Requirements: The Key to Sustainability

Becker, Christoph ; Betz, Stefanie ; Chitchyan, Ruzanna ; Duboc, Leticia ; Easterbrook, Steve M ; Penzenstadler, Birgit ; Seyff, Norbert ; Venters, Colin C

Abstract: Software's critical role in society demands a paradigm shift in the software engineering mindset. This shift's focus begins in requirements engineering. This article is part of a special issue on the Future of Software Engineering.

DOI: https://doi.org/10.1109/MS.2015.158

Requirements: The Key to Sustainability

Christoph Becker, Stefanie Betz, Ruzanna Chitchyan, Leticia Duboc, Steve M. Easterbrook, Birgit Penzenstadler, Norbert Seyff, and Colin C. Venters

***Abstract*— The critical role that software plays in society demands a paradigm shift in the mindset of Software Engineering. The focus of this shift begins in Requirements Engineering.**

***Keywords*— Software Engineering, Requirements, Sustainability, Sustainability Design**

I. Introduction

Software systems are a major driver of social and economic activity. Software Engineering (SE) tends to focus on the technical elements - artificial systems with clear boundaries and identifiable parts and connections, modules and dependencies. But software systems are embedded in other technical systems, and in socio-economic and natural systems. This embedding is obvious when the interaction is explicit, such as environmental monitoring or flight control software. However, software-intensive systems have become such an essential part of the fabric of social systems that the boundaries and interactions of the resulting socio-technical systems are often hard to identify. For example, communication, travel booking or procurement systems influence the socio-economic and natural environment through far-reaching effects on how we form relationships, how we travel, and what we buy. These effects are rarely made explicit in the engineering process. The lack of visibility of these effects makes it hard to assess the long-term and cumulative impacts of a software system. Designing for sustainability is a major challenge that can profoundly change the role of software engineering in society. But what does it mean to establish sustainability as a major concern in SE? We argue that as software engineers, we are responsible for the long-term consequences of our software irrespective of the primary purpose of the system under design. In this paper, we focus on requirements as the key leverage point for practitioners who want to develop sustainable software-intensive systems. We use a case adapted from a real-world software project to provide examples for the changes needed in SE, and show how considering sustainability explicitly will affect requirements activities.

II. Sustainability in Software Engineering

Sustainability is the capacity to endure, so the sustainability of a system describes how well this system will continue to exist and function, even as circumstances may change. Sustainability has often been equated with environmental issues, but it is increasingly clear that it requires simultaneous consideration of environmental resources, societal and individual well-being, economic prosperity, and long-term viability of technical infrastructure.

Sustainability of a technical system is very different from the sustainability of a socio-economic system. Software engineers tend to focus on the technical dimension of sustainability, where it is simply a measure of the software system's longevity [1]. However, to understand broader sustainability issues, we need to ask which system to sustain, for whom, over which time frame, and at what cost [2]. Five interrelated dimensions must be considered [3]:

- The **individual** dimension covers individual freedom and agency (the ability to act in an environment), human dignity and fulfillment. It includes the ability of individuals to thrive, exercise their rights and develop freely.
- The **social** dimension covers relationships between individuals and groups. For example, this aspect covers the structures of mutual trust and communication in a social system and the balance between conflicting interests.
- The **economic** dimension covers financial aspects and business value. It includes capital growth and liquidity, questions of investment, and financial operations.
- The **technical** dimension covers the ability to maintain and evolve artificial systems (such as software) over time. It refers to maintenance and evolution, resilience, and the ease of system transitions.
- The **environmental** dimension covers the use of and stewardship of natural resources. It includes questions ranging from immediate waste production and energy consumption to the balance of local ecosystems and concerns of climate change.

Complex software-intensive systems can affect sustainability in any of these dimensions. Changes in one system, in one dimension, often have impacts in other dimensions and other systems. For example, consider a software system that is hard to maintain (technical sustainability). Excessive maintenance costs affect the financial liquidity of the owning company (a social and economic system). This may limit its growth and even threaten its survival (economic sustainability).

Similar trade-offs occur across other dimensions. For example, carbon offsets incentivize environmentally

sustainable behaviour through trade-offs with the economic dimension. The triple bottom line perspective [4] requires a business to account for social and environmental as well as financial outcomes. The corresponding business practices have led to a surge in the number of social enterprises, which achieve survival rates above average for newly-founded businesses [5].

Increasingly, software engineers need to understand the effects by which decisions taken in the design of software systems can enable or undermine sustainability of socio-economic and natural systems over time (see sidebar 1). Since the concept of sustainability is inherently multidisciplinary, any effort to define sustainability involves concepts, principles, and methods from a range of disciplines and makes an integrated view crucial for an effective systems design process. The notion of sustainability design brings these concerns together using systems thinking principles (see sidebar 2) [6].

III. Requirements: A tale of two projects

The impact a software system will have on its environment is often determined by how the software engineers understand its requirements. The foundation of this impact is set in the decisions on which system to build (if any at all); in the choices of whom to ask and whom to involve, and in the specification of what constitutes success.

The following example illustrates how requirements activities are usually carried out. It describes a procurement system that supports the process of purchasing products and contracting services in a private company in the energy sector. Products, services and suppliers must pass the company's approval process and be registered in the system prior to a purchase. This approval considers the supplier's reliability and capacity to deliver, and in some cases, adherence to international standards of environmental management, health and safety management.

The example is inspired by a real-world case studied by one of the authors [7]. The basis for our example is taken from this case; the description is adapted to be representative of what typically happens in software projects. Further below we show how a commitment to sustainability changes the project.

A. As it often happens: System development without sustainability design

The project **purpose** is to maximize the procurement efficiency of the organization, increase financial return, and ensure suppliers' compliance with certain rules. The criteria for selecting products and services focus on price, delivery time and payment conditions. Using a **stakeholder** influence matrix, the project leader focuses on those stakeholders who can `stop the show'. The project **scope** is determined by a few influential stakeholders early on, so that the project can focus on minimal design scope in order to maximize project speed. The project team moves swiftly to determine the **boundaries** of the software to be, and the only scoping questions revolve around the software's interfaces with neighboring systems.

The **success** criteria for the project are to develop and deliver the system within the given budget and time. The question of feasibility centers on the expected amortization period of the software project investment. Risk analysis focuses on economic risks that could inhibit project completion.

Sidebar 1: Classifying the systemic effects of software.

Many critical effects that occur in socio-technical systems play out over time, so we need to consider not just immediate features and effects of our systems, but longer-running, aggregate and cumulative impact. We distinguish three orders of effects, adapted from [8]:

Immediate effects are direct effects of the production, use and disposal of software systems. This includes the immediate benefit of system features and the full lifecycle impacts, as would be included in a Life-Cycle Assessment (LCA) approach, which evaluates the environmental impact of a product's life from the extraction of raw materials to its disposal or recycling.

Enabling effects arise from the application of a system over time. This includes opportunities to consume more (or less) resources, but also other changes induced by the usage of a system.

Structural effects represent "persistent changes observable at the macro level. Structures emerge from the entirety of actions at the micro level and, in turn, influence these actions" [8]. Ongoing use of a new software system can lead to shifts in accumulation of capital, drive changes in social norms, policies and laws, and alter our relationship with the natural world.

Consider the *airbnb.com* service. Its immediate effects include resources consumed and jobs created during its development, energy consumed during its deployment, and the room renting and booking services it offers. Its enabling effects include changes in how its users make travel arrangements as alternatives to hotel bookings, and how property owners rent out space. These enabling effects (the so-called "sharing economy") have been alternatively praised and criticized for their far-reaching structural impacts. For example, airbnb represents a substantial share of the buy-to-let market in major cities, and the continuing price surges in the hot-spots of these cities have been linked to the density of buy-to-let properties. Many of these exist only because of the arbitrage provided by services such as airbnb: The system enables transactions that provide higher return on investment than long-term rentals. This has caused major concerns in several large cities.

Requirements **elicitation** requests input from the stakeholders through structured forms to identify what they want the system to do. Additionally, previous systems are analyzed and business process documents consulted. Requirements **prioritization** is determined by functional requirements and economic constraints and completed quickly, as the core stakeholder group has strong consensus. The requirements specification is **documented** following the IEEE 830 Requirements Specifications Template. System **measurement and monitoring** uses indicators about performance and availability. The system is completed on time and within budget and shows a reasonably low rate of faults, so the project is considered a **success** at completion.

B. As it can happen: System development practicing sustainability design

Consider conducting the same project with a commitment to treating sustainability as a first-class concern in line with the principles of sustainability design (sidebar 2).

Sidebar 2: Sustainability principles for Software Engineering [6]

1. Sustainability is **systemic**; the system under consideration can never be treated in isolation from its environment.
2. Sustainability is **multi-dimensional**; five key dimensions are economic, social, environmental, technical, and individual sustainability.

3. Sustainability is **inter-disciplinary**; sustainability design in SE requires appreciation of concepts from other disciplines and must work across multiple disciplines.
4. Sustainability **transcends** the purpose of the software; any software that is intended to be used can impact the sustainability of its containing socio-economic, sociotechnical, cultural and natural environments.
5. Sustainability is **multi-level**; it requires us to consider at least two spheres in the system design process: the system under design and its sustainability, and the wider system of which it will be part.
6. Sustainability is **multi-opportunity**; it requires us to seek interventions that have the most leverage on a system [9] and consider the opportunity costs.
7. Sustainability is **multi-timescaled**; long-term thinking is required to address the multiple timescales on which sustainability effects take place.
8. Sustainability is **non-zero-sum**; changing the design of a system to consider the long-term effects does not automatically imply making sacrifices in the present.
9. System **visibility** is a necessary precondition and enabler for sustainability design because only a transparent status of the system and its context, made visible at different levels of abstraction and perspectives, can enable informed responsible choices of system designers.

See www.sustainabilitydesign.org and [6].

When the **purpose** of the project is discussed, the initial project team discusses the company's values and responsibilities and identifies opportunities to support the sustainable development of the company. For example, the system can support sustainability in the supply chain by making transparent the carbon footprint of purchases and facilitate the selection of providers who apply sustainable practices. This does not change the overall project objectives, but influences subsequent steps.

The **scope** of requirements analysis starts with an inclusive and integrated view of the procurement processes, material flows into the company, and the social and political environment of the local community. When defining possible system **boundaries**, the team experiments with multiple perspectives and works jointly with the procurement department and others.

They expand the set of **stakeholders** and draw on knowledge beyond the project team by using a stakeholder impact analysis that considers enabling and structural effects to identify those most affected by the project, including those external to the company. Stakeholders include local supplier representatives, service delivery organizations, process analysts, the CTO, and the strategic planning and foresight group.

To keep the number of stakeholders manageable, a sustainability expert acts as a surrogate stakeholder for others in the community and the further environment that may be affected by the system. A team member is assigned to each of the five sustainability dimensions, so that responsibility for identifying possible effects is clear and effective communication with additional stakeholders can take place. These team members consult relevant experts in areas such as supply chain sustainability, carbon accounting, socially responsible procurement, and anthropologists analyzing and interpreting current technological developments and its impact on our societies.

The team agrees that the **success criteria** of the project are not restricted to whether it is delivered on time and within budget, but will be measured and monitored **over a period of 36 months after project completion**. In this period, a set of indicators will be measured that cover the five dimensions of sustainability. The team will attempt to measure technical debt, social reputation and the improvement of relations with the local community, individual aspects such as privacy compliance and the satisfaction of those involved in the procurement process, environmental aspects such as the total carbon footprint of the products and services acquired, and amortization of the project costs and improved cost-benefit relations in procurement.

During **risk** analysis, the team considers internal and external risks related to systemic effects in all five dimensions. For example, considering the evolving regulations on environmental accountability as a risk, the team develops a set of transparency requirements for the system. They also identify uncertainties about future shifts in procurement as sustainable products become more competitive. As a result, they include a feature to monitor these uncertainties.

During requirements **elicitation**, participatory techniques are employed, and the inclusive perspective enables the project to leverage contributions from a broader set of stakeholders, including local service providers. In a series of workshops, they use a sustainability reference goal model to derive specific sustainability goals for their project and align them with other system goals, while deriving extended usage scenarios with the local community representatives.

The resulting requirements **document** is based on a template that includes checklists for sustainability criteria and standards compliance in all five dimensions. The document is circulated among all stakeholders, and is shared with regulatory agencies to demonstrate the project meets relevant sustainability rules. As a result, it is also used more actively in subsequent stages.

IV. Sustainability Debt

The system that results from this procurement project is different when sustainability principles and therefore long-term consequences are considered.

Focusing on sustainability design, software engineers have to adopt a mindset quite different from the puzzle-solving attitude often found in engineering and business. The objective is to identify and understand "wicked problems": problems that are deeply embedded in a complex system with no definitive formulation, and no clear stopping rule. In such cases, every solution changes the nature of the problem, so there is little opportunity for learning through trial and error [10]. What is needed, instead, is an adaptive, responsive, iterative approach that emphasizes shared understanding.

Figure 1 highlights selected direct, enabling and structural effects of the procurement system in the five sustainability dimensions. Consider a system feature that tracks the carbon footprint of individual products. The feature enables users to choose products with lower carbon footprints. The compound structural effect in the economic dimension can benefit local

suppliers with environmentally sustainable production and lead to an overall reduction of the carbon footprint.

The diagram serves as a visual aid to support interactive collaboration among stakeholders to discover, document and validate potential effects of the system. Not all effects will be positive: For example, automating product selection rules to minimize carbon footprint takes away the freedom of the manager to take decisions in the procurement process [11]. This can reduce mutual trust between members of the organization.

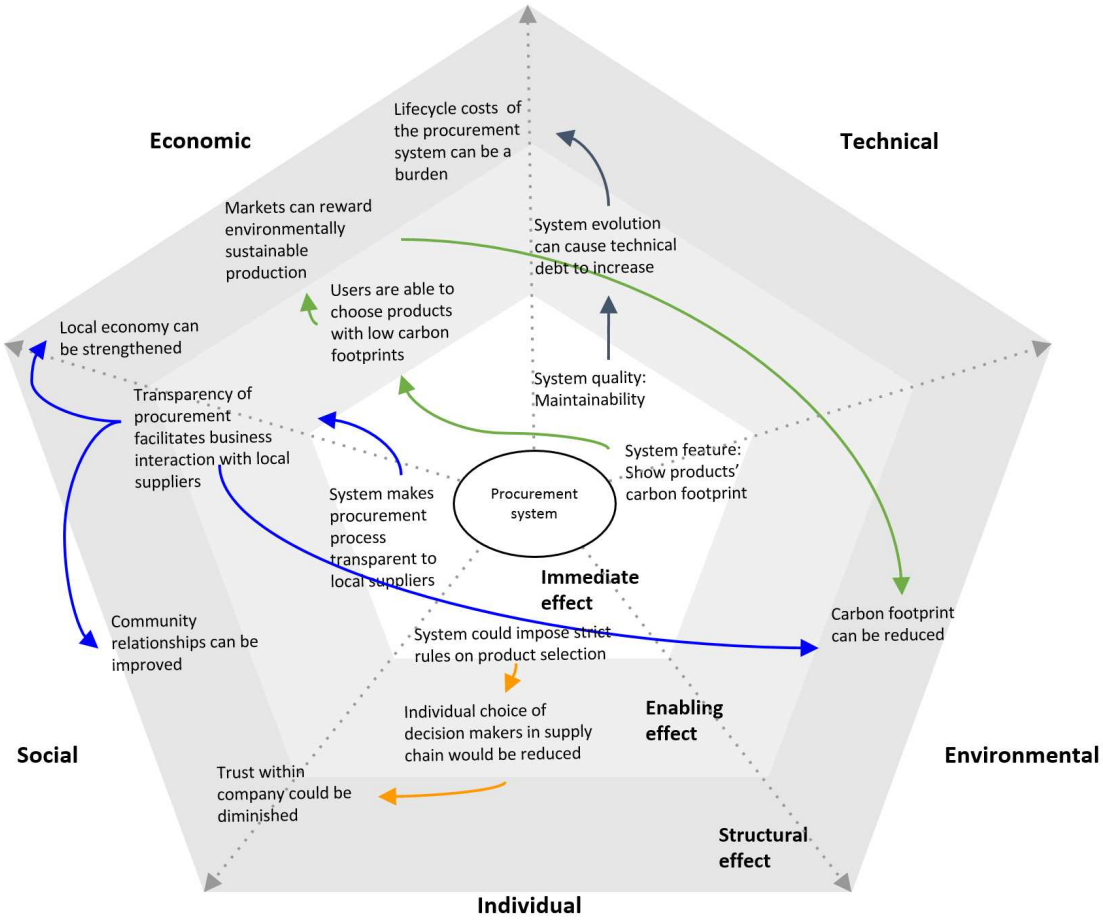

Figure 1. Selected systemic effects of the procurement system

The diagram also facilitates a conversation about ``sustainability debt'' [12]: the invisible effects of taking decisions for the present that accumulate over time in each of the five dimensions. When we increase energy consumption, reduce individual privacy, impose technical barriers, or incur additional financial costs, we incur debts in these dimensions towards different stakeholders. Making these effects visible is the first step to understanding and considering them in systems design decisions.

V. Requirements are the Key

In the tale of two projects, we have seen a series of decision points in the process of designing a system. Many of these are requirements engineering activities that will occur repeatedly in all iterations throughout the project. Each decision influences the decision space of subsequent choices and has a profound impact on the system to be designed and the effects it will have. Table 1 highlights how key activities change when we consider sustainability design principles.

The leverage of requirements becomes clear when we consider their relationships with engineering techniques. We develop techniques in order to quantify, construct, and test artifacts and to control whether the results fall within an acceptable range. However, for design concerns such as usability, performance, maintainability, or sustainability, such technique are only applied once a need has been identified. Without such a need, the engineering techniques will remain unused, and hence have no effect on the project. For example, techniques for increasing technical sustainability abound, ranging from architectural design patterns to documentation guidelines. Yet, since applying these techniques often involves an upfront investment of effort, it is only done when a longer life expectancy of a system is recognized and expressed. On the other hand, a stated requirement for which no current technique exists will lead to an identified gap in technological ability. This means that in practice, systemic changes to the activities in Table 1 will dominate the effects of whatever techniques we develop to support these activities.

Requirement engineers therefore play a key role in sustainability. As ``sustainability engineers'', they go beyond a

narrow system perspective and follow an interdisciplinary, systems-oriented, stakeholder-focused approach, supported by higher management and executives. Their task is to understand the nature of software-intensive systems and the impact those can have on their social, technical, economic and natural environment and the individuals in that environment.

This responsibility is reflected in the new UK Standard for Professional Engineering Competence (UK-SPEC), which

Table 1 SE practices for sustainability

Task	Standard current practice	Future practice focuses on
Mindsetting	The world is a puzzle, and we should "solve the problem"	The world is complex, and we should first "understand the dilemmas".
Project objective, System purpose and boundary scoping	Focus on the immediate business need and key system features. Do not question the purpose of the project or the purpose of the system.	Emphasize the effects that the project can have on sustainability in all dimensions. Strive to advance sustainability in multiple dimensions simultaneously. Experiment with different system boundaries to understand the difference this might cause in its impact.
External constraints identification	See constraints as imposed by the direct environment of the system and its technical interfaces. Minimize the constraints considered, but include legal, safety, security, technical, and business resources.	See constraints in each dimension as opportunities. Look for constraints from additional sources, starting with company Corporate Social Responsibility policies, legislation and standards for sustainability.
Stakeholder identification	Minimize the number of stakeholders involved and focus on stakeholders who have influence. Focus on internal stakeholders and exclude unreachable stakeholders.	Maximize stakeholder involvement in an inclusive perspective integrating external stakeholders and involve those who are affected. Assign a dedicated role to be responsible for sustainability and introduce surrogate stakeholders to represent outside interests.
Success criteria definition	Focus on the financial bottom line at project completion. Measure business outcome and financial return on investment.	Focus on advancing multiple dimensions simultaneously, including financial aspects, and take into account that most of the effects occur after project completion.
Requirements Elicitation	Focus on the features and the immediate effects the stakeholders want.	Help stakeholders to understand the enabling effects the system will have. Use creativity techniques and long-term scenarios to forecast potential structural impact.
Risk identification	Identify risks that threaten timely project completion within budget.	Include effects on the system's wider environment. Include enabling and structural effects and risks that can develop over time.
Trade-off analysis	View it as a prioritization and selection problem and let the key stakeholders decide.	Strive to transform sustainability trade-offs into mutually beneficial situations. Make sure that sustainability trade-offs are discussed by a wider range of stakeholders (or their surrogates).
Go/No-Go decision	Base the decision on feasibility, financial cost/benefit and risk exposure to project participants, i.e. internal stakeholders.	This continues to be an internal business decision, but is documented to show to external audiences that sustainability indicators and enabling effects were taken into account. The decision is based on a consideration of positive and negative effects on all five dimensions.
Requirements validation	Let key stakeholders verify that their interests are captured.	Ensure broad community involvement focused on understanding effects.
Project completion	Verify whether success criteria are met on completion date. After that, focus on maintenance and evolution.	Evaluate the effects on all five dimensions over a certain timeframe after completion aligned with the expected timescale of effects.
Requirements documentation	Current templates ignore long-term effects and sustainability considerations.	Templates require information about sustainability as a design concern and support analysts with checklists.

explicitly defines the role of engineers such that they shall ``Act in accordance with the principles of sustainability, and prevent avoidable adverse impact on the environment and society." [13]. It is up to SE curricula developers to equip future software engineers with the competences required to simultaneously advance goals in all five dimensions, beyond the technical and economic.

For a long time, concerns about such effects have taken a backseat in SE, but this is changing as standards are being adjusted. For example, the working group WG42 on ISO/IEC 42030 (Architecture evaluation) is discussing energy efficiency and environmental concerns at the software

architecture level and the IEEE P1680.1 for Environmental Assessment of PC products is being revised.

While this is an important step, a full consideration of all five sustainability dimensions is needed on the level of quality models, systems documentation templates, and the analysis of systemic effects throughout system lifecycle stages. It will often be the responsibility of the requirements engineers to introduce relevant standards in each of the five dimensions into the elicitation and specification process. To support this, sustainability considerations related to quality attributes of software systems in use should be integrated into revisions of the ISO 25000 series, while ISO 29148 should acknowledge the importance of system characteristics beyond the interaction with human users and encourage consideration of the systemic effects of software systems in RE.

VI. Software Engineering in Society

The critical role that software plays in society demands a paradigm shift in the mindset of SE. Sustainability design emphasizes an appreciation of 'wicked problems' over a focus on puzzles and pieces; systems thinking over computational problem solving; and an integrated understanding of systems over a divide-and-conquer approach to systems analysis.

While these are challenging shifts that do not come easy, taking such perspectives provides an opportunity to stand out, an invitation to innovate, and an occasion for software engineers and companies to distinguish themselves with a unique selling point in a competitive market. We also have an opportunity to help shape broader sustainability policy. A shift to a sustainable society requires both large-scale change in government policy and a change in engineering and business practice; neither on their own will suffice. But regulatory change is much easier if it builds on established best practice, so software practitioners need to lead the way.

If you agree that we, as software engineers, have a responsibility for the long-term impacts of the systems we design, the principles of sustainability design provide an opportunity to get started. We can and should start now, and practitioners can lead the way: We need to collect experiences in applying sustainability principles in software engineering and learn from the process. An important way to make this vision of software as a force for sustainability a reality is by cooperation between industry and academia.

Successful collaborations to integrate sustainability concerns into established practices can have significant impact on the long-term effects of the systems we design. To facilitate this, we must:

- Identify and tackle causes of unsustainable software design. For this, industry can invite academics to research, analyze, and re-engineer their current development processes and practices for improved sustainability;
- Develop a number of exemplar case studies that demonstrate the benefits of sustainability design in software engineering. For this, early-adopter industrial collaborators can partner with academics to apply research findings such as those summarized in Table 1 and report on longer term results;
- Build competencies in the theory and practice of sustainable design into the training of all software engineers. Industry can make the demand for software practitioners trained in sustainability principles explicit by requiring specific competences from potential employees. Researchers and educators should develop improved curricula that incorporate sustainability principles and ensure that future software professional possess the competences needed to advance sustainability goals through SE.

Let's get started.

VII. Acknowledgments

This work is supported by DFG EnviroSiSE (PE2044/1-1), FAPERJ (210.551/2015), CNPQ (14/2014), NSERC (RGPIN-2014-06638), the European Social Fund, Ministry of Science, Research and the Arts Baden-Württemberg, and WWTF through project BenchmarkDP (ICT2012-46).

Special thanks to our friend and colleague Sedef Akinli Kocak, PhD researcher at Ryerson University in Toronto, for her contributions to this article.

Highlights

- The critical role that software plays in society demands a paradigm shift in the mindset of Software Engineering.
- Sustainability design favors integrated understanding over a divide-and-conquer approach to systems analysis.
- Sustainability Design requires an appreciation of 'wicked problems' in Requirements Engineering.
- Integrating sustainability concerns can significantly impact the long-term effects of the systems we design.
- Sustainability design provides an opportunity for software companies to stand out with a unique value proposition.

References

[1] H. Koziolek, "Sustainability evaluation of software architectures: a systematic review," in Proc. of QoSA-ISARCS'11. ACM, 2011, pp. 3–12.

[2] J. A. Tainter, "Social complexity and sustainability," Ecological Complexity, vol. 3, no. 2, pp. 91–103, 2006.

[3] B. Penzenstadler, A. Raturi, D. Richardson, and B. Tomlinson, "Safety, security, now sustainability: The nonfunctional requirement for the 21st century," IEEE Software, vol. 31, no. 3, pp. 40–47, 2014.

[4] J. Elkington, "Enter the triple bottom line," The triple bottom line: Does it all add up, pp. 1–16, 2004.

[5] E3M, "Who Lives the Longest? Busting the Social Venture Survival Myth." [Online]. Available: http://socialbusinessint.com/wp-content/uploads/Who-lives-the-longest_-FINAL-version2.pdf

[6] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters, "Sustainability Design and Software: The Karlskrona Manifesto," in Proc. 2015 Int'l Conf. Software Eng. (ICSE'15), 2015.

[7] C. Bomfim, W. Nunes, L. Duboc, and M. Schots, "Modelling sustainability in a procurement system: An experience report," in Proc. 2014 Requirements Engineering (RE'14). IEEE, 2014, pp. 402–411.
[8] L. M. Hilty and B. Aebischer, "ICT for sustainability: An emerging research field," in ICT Innovations for Sustainability. Springer, 2015, pp. 3–36.
[9] D. H. Meadows, Leverage points: Places to intervene in a system. Sustainability Institute Hartland, VT, 1999.
[10] J. A. Klein, "A reexamination of autonomy in light of new manufacturing practices," Human Relations, vol. 44, no. 1, pp. 21–38, 1991.
[11] S. Betz, C. Becker, R. Chitchyan, L. Duboc, S. M. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters, "Sustainability debt: A metaphor to support sustainability design decisions," in Proc. RE4SuSy 2015. http://ceur-ws.org/Vol-1416/, 2015.
[12] UK Standard for Professional Engineering Competence (UK-SPEC). The Engineering Council, 2014.

Christoph Becker is an Assistant Professor at the University of Toronto, where he leads the Digital Curation Institute, and a Senior Scientist at the Vienna University of Technology in Austria. His research focuses on sustainability in software engineering and information systems design; digital curation and digital preservation; and digital libraries. Becker received his PhD in computer science from the Vienna University of Technology. Contact him at christoph.becker@utoronto.ca.

Stefanie Betz is a senior research scientist at the Department of Applied Informatics and Formal Description Methods, Karlsruhe Institute of Technology, Germany. Her research is centered on sustainable software and systems engineering, particularly from the perspective of requirements engineering and business process management. Betz received her PhD in Applied Informatics at Karlsruhe Institute of Technology. Contact her at stefanie.betz@kit.edu.

Ruzanna Chitchyan is a lecturer at the Department of Computer Science, University of Leicester, UK and a member of the Centre for Landscape and Climate Research. Her research is centred on requirements engineering and architecture design for software-intensive socio-technical systems engineering and sustainability. Chitchyan received her PhD in Software Engineering from the Lancaster University, UK. Contact her at rc256@leicester.ac.uk.

Leticia Duboc Leticia Duboc is a lecturer at the State University of Rio de Janeiro, Brazil and an Honorary Research Fellow at the University of Birmingham, UK. Her research focuses on sustainability and scalability of software systems, particularly from the perspective of requirements engineering and early analysis of software qualities. Duboc received her PhD in computer science from the University College London UK. Contact her at leticia@ime.uerj.br.

Steve M. Easterbrook is a professor at the University of Toronto and a member of Centre for Environment and Centre for Global Change Science. His research focuses on climate informatics, and more specifically, the applications of computer science and software engineering to the challenge posed by global climate change. Easterbrook received his PhD in Computing from Imperial College, London. Contact him at sme@cs.toronto.edu.

Birgit Penzenstadler is an Assistant Professor of software engineering at the California State University, Long Beach. Her research focusses on software engineering for sustainability and resilience and she leads the Resilience Lab at CSULB. Penzenstadler received her PhD and a habilitation degree from the Technical University of Munich, Germany. Contact her at birgit.penzenstadler@csulb.edu.

Norbert Seyff is a professor at the University of Applied Sciences and Arts Northwestern Switzerland and a senior research associate at the University of Zurich. His current research focus on requirements engineering and software modeling. He has a particular interest in empowering and supporting end-users participation in system development. Seyff received his PhD in computer science from Johannes Kepler University Linz, Austria. Contact him at norbert.seyff@fhnw.ch.

Colin C. Venters is a senior lecturer in Software Systems Engineering at the University of Huddersfield, UK. His current research focuses on sustainable software systems engineering from a software architecture perspective for pre-system understanding and post-system maintenance and evolution. Venters received his PhD in computer science from the University of Manchester, UK. Contact him at c.venters@hud.ac.uk.